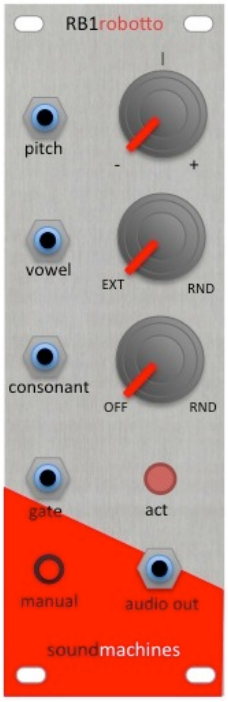
# soundmachines




# RB1robotto User's Manual

- Introduction
- Care and feeding
- Specifications
- Hackers only chapter

## Introduction

Robotto is a Voltage Controlled Voice, a  singing module, a nice addition to any modular synthesizer.

Built around the Soundgin® (now Babblebot®) chip and an Arduino® microcontroller, Robotto is basically a controllable vocalizer.

The parameters that the user can modify through a CV/Gate interface are the pitch, the vowel and the consonant.

Not all the 0-5V pitch interval is useful, as the synthesizer core inside the Babbleboth is doing a good job around the central octaves. Anyway, if oddness is what you want, feel free to explore.

## Care and Feeding

RB1robotto is an 8hp eurorack module designed to comply with the Doepfer® power standard. As the module has unipolar inputs (and an AC coupled audio output) we just need the 12V and GND. 5V is generated internally.

Please use the supplied power cable. The red conductor on the cable is connected to -12V supply rail as the standard implies.

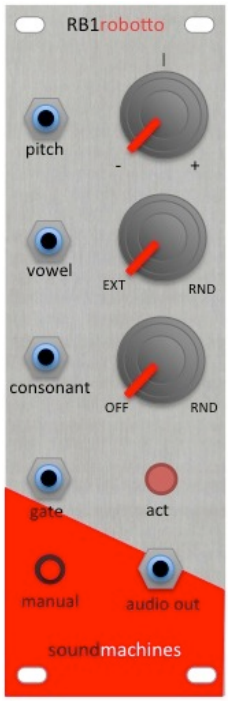
On the PCB of RB1robotto there is the clear indication of the polarity of the connector so, please, if you have a module without the connector inserted, take care of the just orientation. RB1robotto is protected against reverse insertion but you'll never know…

All the inputs are protected from over and undervoltages. Specifically, pitch and gate inputs are protected up to 5V and the others inputs up to 12V.

Current rating: the module requires XXXmA on the +12V rail. No current is drawed on the negative supply line.

## Specifications

Robotto module is a sound generating module. Basic usage could be viewed as a very simple synth voice. By using pitch cv and gate inputs, the module will sing to the user's need. Please refer to the following figure for input/output jacks and control knobs:




"**pitch**" input: this input tracks the pitch of the 'voice' outputting from the audio out jack of the module. The useful input voltage is from 0 to 5V. The tracking theoretically could be 5 octaves but, speaking (pun intended) of voices, it makes much more sense to limit the tone to the three central octaves. The pitch - / + knob allow the user to tune or detune the robotto to the incoming voltage.

**Pitch knob**: this knob,serves as a fine tune knob. It is mostly useful to adapt the tracking to the incoming voltage. Center position (detent) should be good with correctly tuned voltages.

"**vowel**" input: this input jack permits the user to select the vowel of the voice. The vowel could be changed also after the gate signal goes high, thus permitting to change the vowel during the playing of a voice tone. The vowel input is functional only when the corresponding knob is set to 'ext'. The input range of the input is from 0 to 10V (though the input is 12V 'tolerant'). The range is divided equally to the 5 main vowels (aa, e, i, or, ue).

**Vowel knob**: the knob is used to set the vowel and a couple of functions. When the knob is set to ext, the CV will determine the vowel that's been played. When the knob is set to rnd, random vowels will be chosen every time a gate transition willbe activated (either manual or by means of external connections. When the knob is in the active area (a *little more*® than EXT and a *little less*® than RND) the user simply changes the vowel by hand.

"**consonant**" input: Same applies as the Vowel section, just, you know, regarding consonants. Also the input has the same range. The consonant that we choose for the robotto are the following, mapped from 0 to 10V: *se,thh,ko,ch,m,j,do*.

**Consonant knob**: if turned fully CCW, in the OFF position, no consonant will be applied to the sound. Note that the consonant pronuciation takes time and it will delay the 'harmonic' part of the sound so we decided to give you a way to disable this. If turned fully CW, to the RND sign, the consonant will be randomized at each GATE trigger. When the consonant knob is in the active range, the consonant is determined either by the pot (if the consonant CV is 0V or the jack is not inserted) or the CV input.

"**gate**" input: this is the trigger for the robotto to sing. Enough said. The RB1robotto engine is retriggerable, as the minimum duration of the sound is very

small (a little  longer when used with consonants). Gate input is standardized to 5V.

"**act**" Led: this indicator is triggered by the generator chip (soundgin) and tells you when it's doing something (i.e. generating speech, sound, noises). It's useful also to understand if the module is correctly used and you actually should expect something at his output!

"**manual'** button: this is a simple manual input (direct) to the gate input.

"**audio out**" jack: this is a secret function. We are not allowed to disclose it yet.

## Hackers Only:

We decided to 'open' the source code fo the  robotto because many of its users would possibly desire to change its way of operating. Considering that the sound source inside (soundgin® / babblebot® chip) is quite powerful, robotto could become something other….. from a complex monophonic synth voice, maybe with an integral arpeggiator or generative stuff, to a chord generator, as the babblebot has six oscillators that could be used as simple voices (just choosing the waveform and the ADSR). Many other things could be (and actually have been) thought of, so we encourage everyone to create their own 'robotto' hack. On the soundmachines site (www.sound-machines.it), in the RB1robotto page, there is the link to the firmware repository, for you to play with (or to restore robotto to its original firmware). Contributors are welcome!

Here some basic information on the software and hardware of the board:

To allow you, dear user/hacker, to work on the module without having to solder, burn or harm anything/anyone, we put a very nice pin-strip connector on top of the module pcb. This connector is compatible, among other, with the version of the FTDI® serial interface sold by Sparkfun (www.sparkfun.com). As of today, the correct product number is DEV-09716. The interface will also give power to the module, so you can test your stuff without having to put the module into the modular!

By having at hand this interface, downloading the Arduino® 1.0 environment from the Arduino® website (www.arduino.cc), and starting some sort of creative mental process, you can start coding and downloading stuff to the module.

DISCLAIMER: there is the remote possibility that you screw-up the Arduino® bootloader that we burnt on the board. No big deal. The board has a connector to reprogram the Atmel® microcontroller. By using another Arduino® board or an even cheaper solution (Atmel® programmer made from some junk that you have lying in your drawers maybe..) you can easily restore the bootloader. Please refer to the Arduino® website for doing this, it's very, very well documented!

There are, apart from the controls that you see on the panel, several unused connection on the microcontroller (that is compatible with Arduino®) that we brought to exposed thru-hole pads on the PCB. Seven of them. Please keep in mind that those are DIRECT connections to the microcontroller's pins, so no protection are in place... As a 'first', and in presence of low voltages (not exceeding 5V) you can just use current limiting resistors (100 ohm) in series with those inputs. As a more robust stuff, protect the input with diodes to 5V and GND.

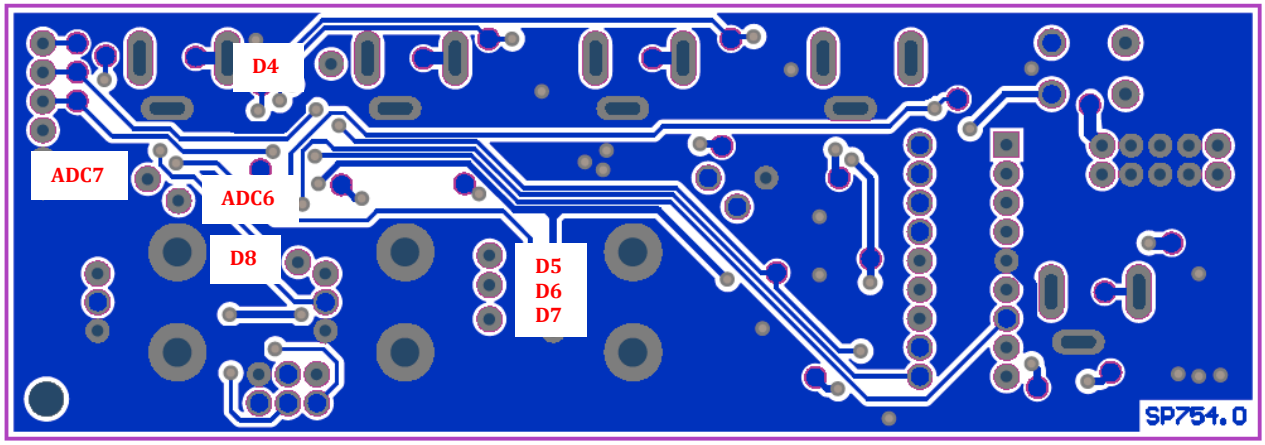
Let's start with the already made connections: By extracting some lines from the arduino sketch, we know that:

```
//Pins definitions: analog
#define PITCH_POT  2
#define VOW_POT    1
#define CONS_POT   0
#define PITCH_CV   3
#define VOW_CV     5
#define CONS_CV    4
//Pins definitions: digital
#define SOUNDGIN_RES    2
#define SOUNDGIN_CTS    3
#define GATE_IN         10
#define TX_SW           9
```

TX_SW is, clearly, the serial line that gives commands to the soundgin/babblebot chip. To find informations (and datasheets) about this chip, direct your browser here: www.babblebot.net

Please refer to the firmware listing to find out what's going on and how to modifiy!

In the figure below you find indications where the unused microcontroller pins are, and what are their name in the Arduino® environment:




So you've got two more analog channels and 5 digital I/O pins. That's pretty much very interesting. Want to connect DACs? Want to connect more pots? You choose....

Enjoy the RB1robotto hacking and please, let us know what you're doing!

*Note: altough this is extremely boring stuff, all the cited trade marks and registered product names are property of their legal owners.*